02810: Introduction to Artificial Intelligence

Lecture 5: Adversarial Search

Nina Gierasimczuk

DTU

Examples of Games

Every day examples of interactions:

- Driving in traffic.
- Bargain-hunting auctioning.
- Governmental elections.

Examples of Games

Every day examples of interactions:

- Driving in traffic.
- Bargain-hunting auctioning.
- Governmental elections.

Historical examples:

- Anthony and Cleopatra.
- Hitler and Stalin.
- Kruschev and Kennedy.

Examples of Games

Every day examples of interactions:

- Driving in traffic.
- Bargain-hunting auctioning.
- Governmental elections.

Historical examples:

- Anthony and Cleopatra.
- Hitler and Stalin.
- Kruschev and Kennedy.

Game theory ‘works’ when agents behave **rationally**.

Game Theory

A branch of applied mathematics used in: social sciences, economics, biology, linguistics, engineering, political science, international relations, computer science, philosophy, etc.

Game Theory

A branch of applied mathematics used in: social sciences, economics, biology, linguistics, engineering, political science, international relations, computer science, philosophy, etc.

Game theory attempts to **mathematically** capture
behavior in **strategic** situations,
in which an individual's success in making choices
depends on the choices of others.

Game Theory

A branch of applied mathematics used in: social sciences, economics, biology, linguistics, engineering, political science, international relations, computer science, philosophy, etc.

Game theory attempts to **mathematically** capture
behavior in **strategic** situations,
in which an individual's success in making choices
depends on the choices of others.

Initially developed for competitive scenarios, later extended to cooperative ones.

Game Theory

A branch of applied mathematics used in: social sciences, economics, biology, linguistics, engineering, political science, international relations, computer science, philosophy, etc.

Game theory attempts to **mathematically** capture
behavior in **strategic** situations,
in which an individual's success in making choices
depends on the choices of others.

Initially developed for competitive scenarios, later extended to cooperative ones.

> *Game theory is a sort of umbrella or 'unified field' theory for the rational side of social science, where 'social' is interpreted broadly, to include human as well as non-human players (computers, animals, plants).*
>
> *(Aumann 1987)*

A Game

A **game** is a description of a situation of interaction of two or more agents.

It includes:

- the players;
- the actions that players can take;
- the preferences of the agents over the possible outcomes of the game.

A Game

A **game** is a description of a situation of interaction of two or more agents.

It includes:

- the players;
- the actions that players can take;
- the preferences of the agents over the possible outcomes of the game.

A **strategic game** is a game in which players involved make one decision each, and make it independently.

Toy Games: Matching Pennies

- Two players: Alice and Bob.
- Both show a coin.
- Bob wins if they show different faces.
- Alice wins if they show the same.
- they each have two strategies: *heads* and *tails*.
- All possible strategy combinations give payoffs for each player.

Toy Games: Matching Pennies (conflict)

Alice	heads	tails
heads	$+$	$-$
tails	$-$	$+$

Bob	**heads**	**tails**
heads	$-$	$+$
tails	$+$	$-$

Toy Games: Matching Pennies (conflict)

Alice	heads	tails
heads	$+$	$-$
tails	$-$	$+$

Bob	**heads**	**tails**
heads	$-$	$+$
tails	$+$	$-$

Together:

	heads	tails
heads	$(+,-)$	$(-,+)$
tails	$(-,+)$	$(+,-)$

Toy Games: Driving Game (cooperation)

Alice	left	right
left	$+$	$-$
right	$-$	$+$

Bob	**left**	**right**
left	$+$	$-$
right	$-$	$+$

Toy Games: Driving Game (cooperation)

Alice	left	right
left	$+$	$-$
right	$-$	$+$

Bob	**left**	**right**
left	$+$	$-$
right	$-$	$+$

Together:

	left	right
left	$(+,+)$	$(-,-)$
right	$(-,-)$	$(+,+)$

Numerical Payoffs

	heads	tails
heads	(1,-1)	(-1,1)
tails	(-1,1)	(1,-1)

	left	right
left	(1,1)	(-1,-1)
right	(-1,-1)	(1,1)

Table: Matching Pennies and Driving Game with payoffs -1 and 1

Any numbers as long as winning>losing.

Zero-sum Games

Recall Matching Pennies game:

	heads	tails
heads	(1,-1)	(-1,1)
tails	(-1,1)	(1,-1)

Zero-sum Games

Recall Matching Pennies game:

	heads	tails
heads	(1,-1)	(-1,1)
tails	(-1,1)	(1,-1)

Payoffs in each cell sum up to 0.
This can be always done for games with pure conflict.

Games in Artificial Intelligence

Games, esp. board games became, the playground for the development of AI.

Games in Artificial Intelligence

Games, esp. board games became, the playground for the development of AI.

The focus on **zero-sum**, **turn-based games**, e.g., Chess or Go.

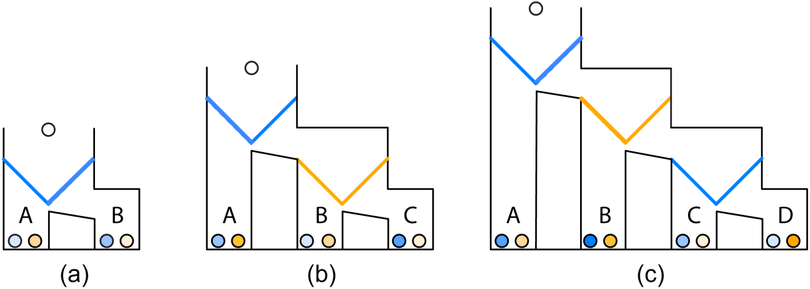
Games in Artificial Intelligence

Games, esp. board games became, the playground for the development of AI.

The focus on **zero-sum**, **turn-based games**, e.g., Chess or Go.

The games of interest are often also **perfect-information** games.

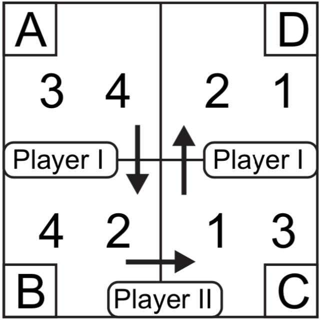
Turn-based Games: Marble Drop Game



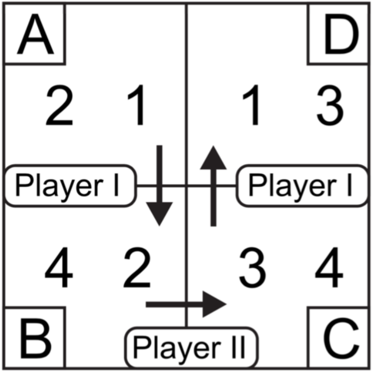
Turn-based Games: Matrix Game

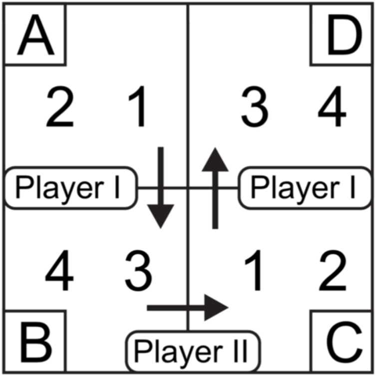
There is one token in the game. It starts being at A. Player I can decide to *move* it to B or let it *stay* in A. If it is moved to B, player II can afterwards *move* it to C or let it *stay* in B. If it is moved to C, player I can *move* it to D or let it *stay* in C. The game ends when a player decides to *stay* or the token ends up in D.

The utility (payoff) of player I and II are the, respectively, left and right numbers in the cell that the token ends up in.



More Matrix Games: Exercise





Search Problems Formally

So far, search problems were defined by:

- **Initial state**.
- $\textsc{Actions}(s)$: **possible actions**.
- $\textsc{Result}(s, a)$: **transition model**.
- **Goal test**.

That induces a **state space** in which we search for a goal state.

Game Problems Formally

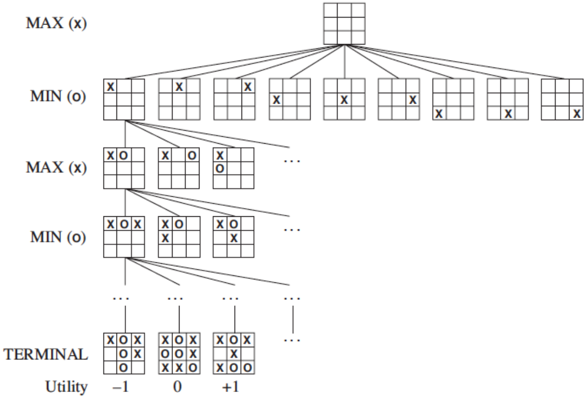
Similarly, games can be described by:

- s_0: **initial state**.
- $\textsc{Player}(s)$: who has the move in s.
- $\textsc{Actions}(s)$: Legal moves in s.
- $\textsc{Result}(s, a)$: **transition model**.
- $\textsc{Terminal-Test}(s)$: **terminal test**. Is the game over?
- $\textsc{Utility}(s, p)$: **utility function** (or **payoff function**).

  Numerical value for player p in terminal state s.

  Example: $+1$ for win and -1 for loose (zero-sum).

Tic-Tac-Toe: Game Tree Example



Minimax

- There are two players: MAX and MIN.
- The game is zero-sum.
- Nodes assigned values representing expected utility for MAX.

Minimax

- There are two players: MAX and MIN.
- The game is zero-sum.
- Nodes assigned values representing expected utility for MAX.

$$\textsc{Minimax}(s) = \begin{cases} \textsc{Utility}(s, \textsc{max}) & \text{if } \textsc{Terminal-Test}(s) \\ \max_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s, a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s, a)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$

In a state s with $\textsc{Player}(s) = \textsc{max}$, the following move is optimal for MAX:

$$\text{argmax}_{a \in \textsc{Actions}(s)} \textsc{Minimax}(s).$$

That is, MAX chooses the move that maximises the minimax value.

Minimax

- There are two players: MAX and MIN.
- The game is zero-sum.
- Nodes assigned values representing expected utility for MAX.

$$\textsc{Minimax}(s) = \begin{cases} \textsc{Utility}(s, \textsc{max}) & \text{if } \textsc{Terminal-Test}(s) \\ \max_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s, a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s, a)) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$

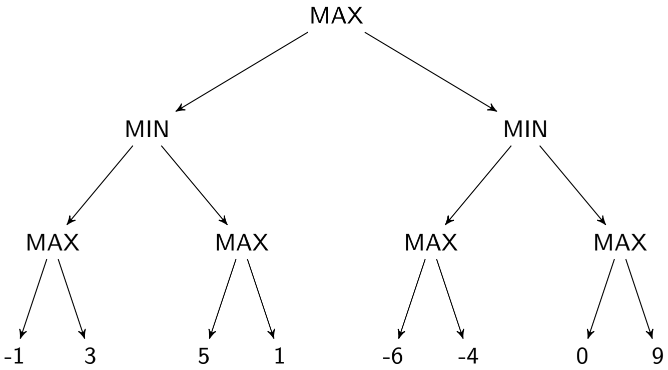
In a state s with $\textsc{Player}(s) = \textsc{max}$, the following move is optimal for MAX:

$$\operatorname{argmax}_{a \in \textsc{Actions}(s)} \textsc{Minimax}(s).$$

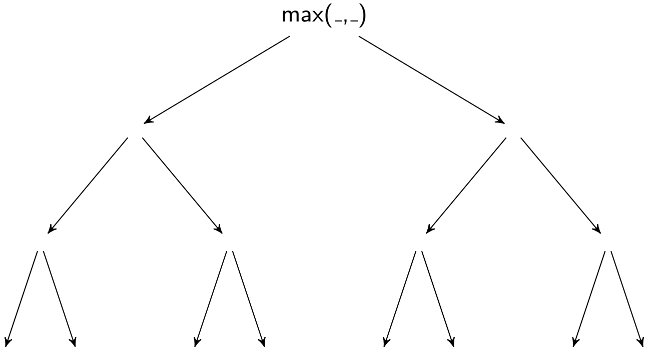
That is, MAX chooses the move that maximises the minimax value.

Is MINIMAX a breadth- or a depth-first search?

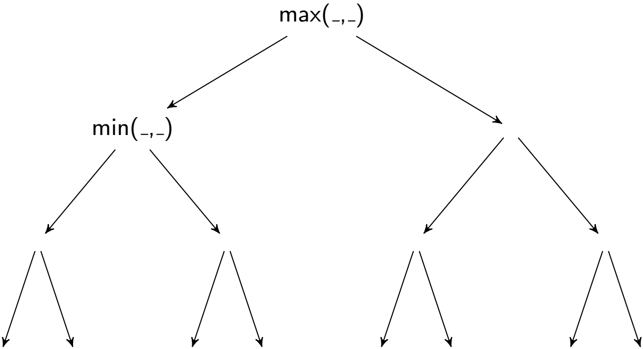
Minimax on an Example



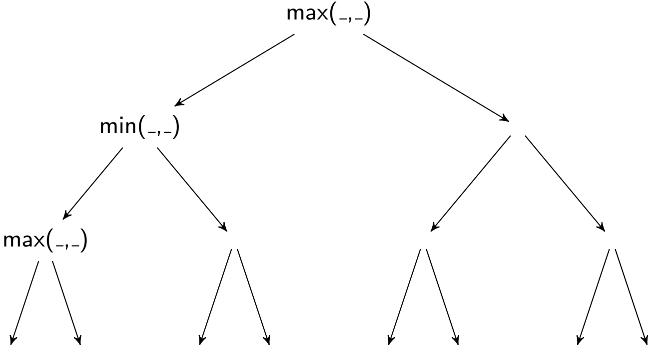
Minimax on an Example



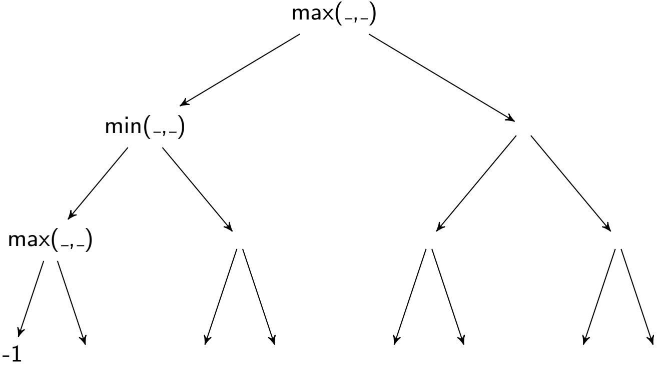
Minimax on an Example



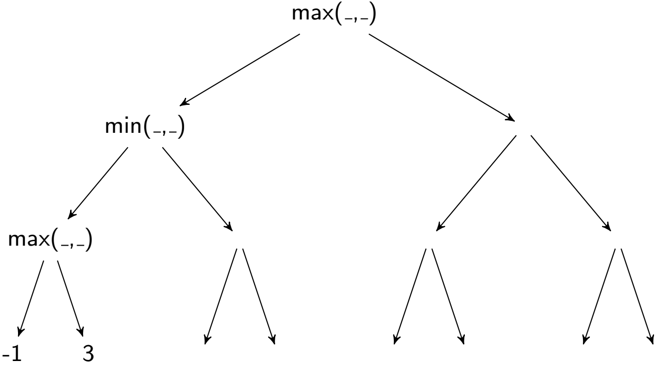
Minimax on an Example



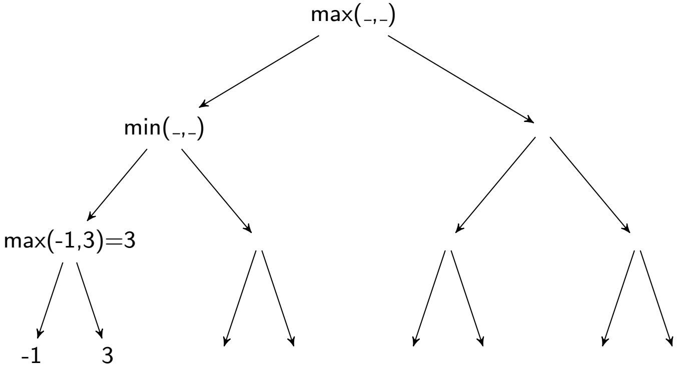
Minimax on an Example



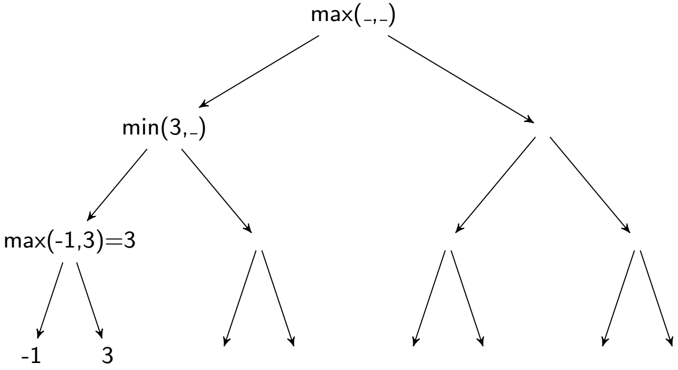
Minimax on an Example



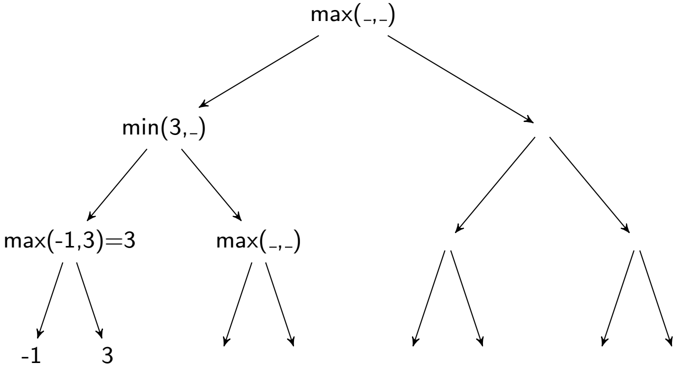
Minimax on an Example



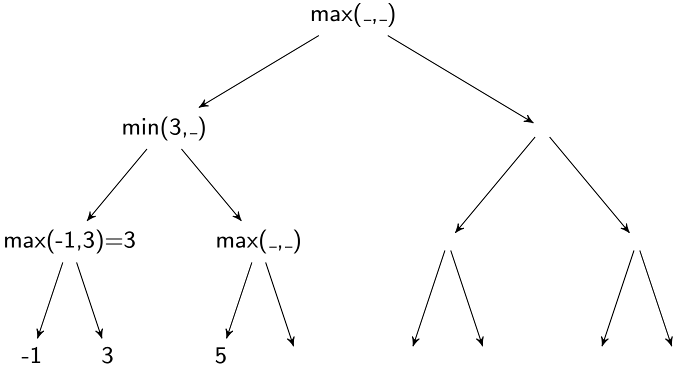
Minimax on an Example



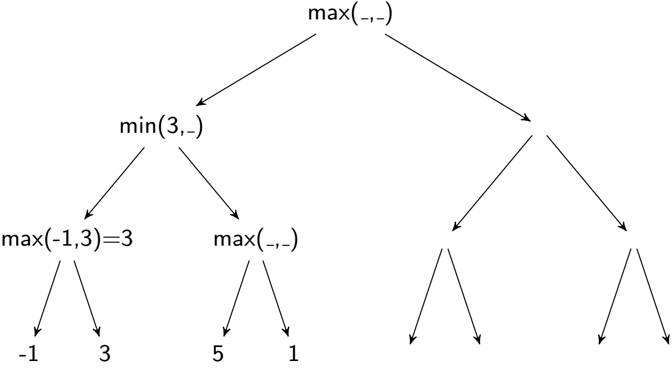
Minimax on an Example



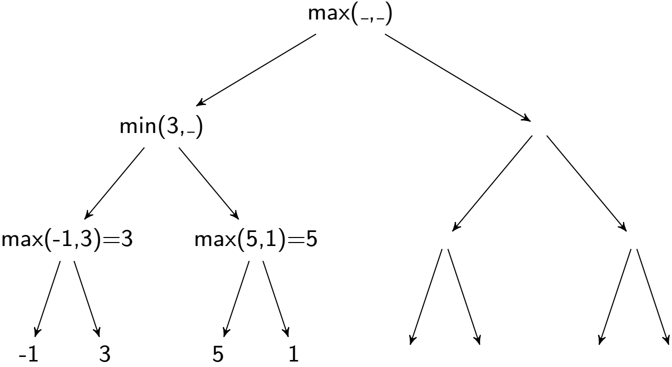
Minimax on an Example



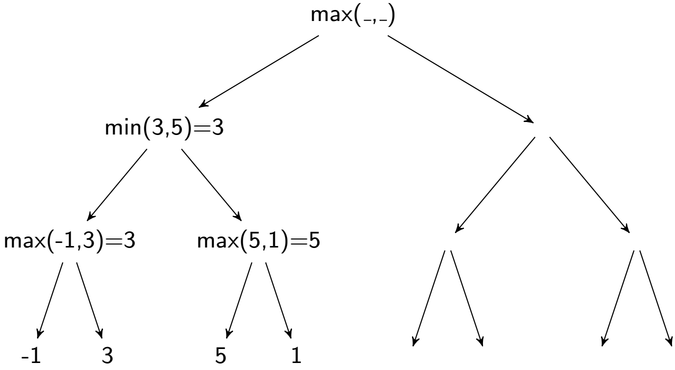
Minimax on an Example



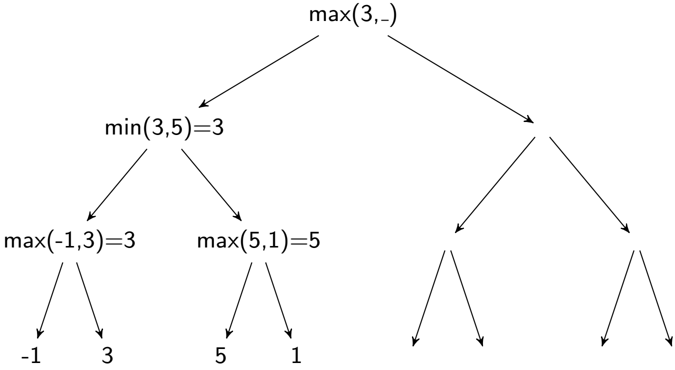
Minimax on an Example



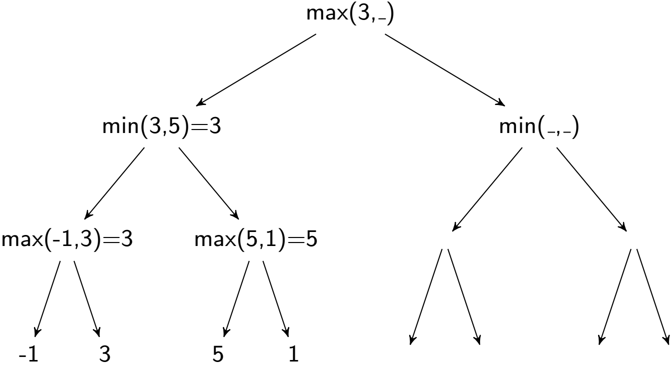
Minimax on an Example



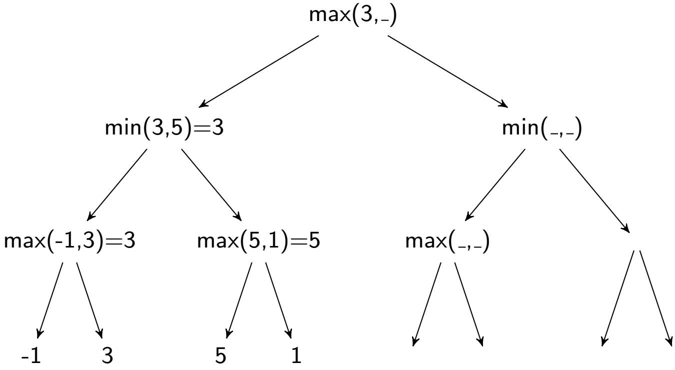
Minimax on an Example



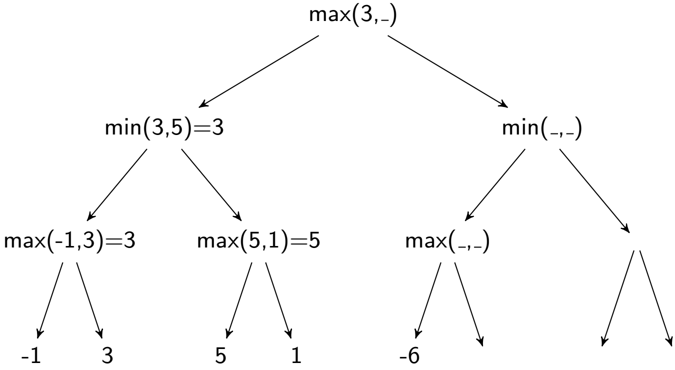
Minimax on an Example



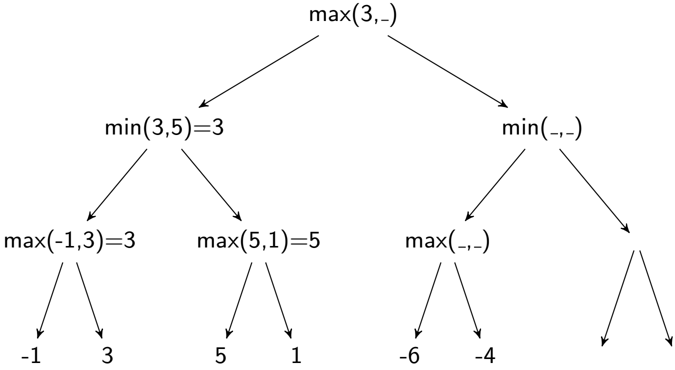
Minimax on an Example



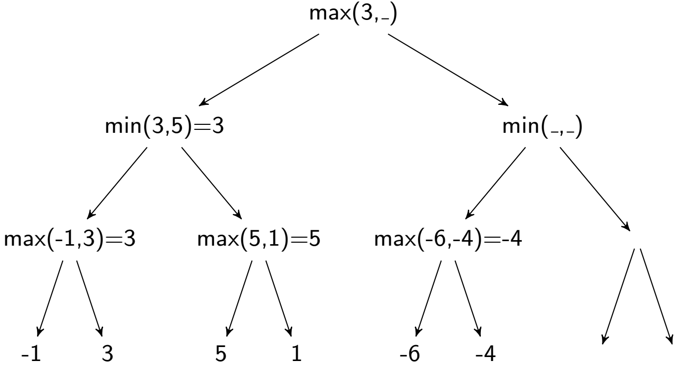
Minimax on an Example



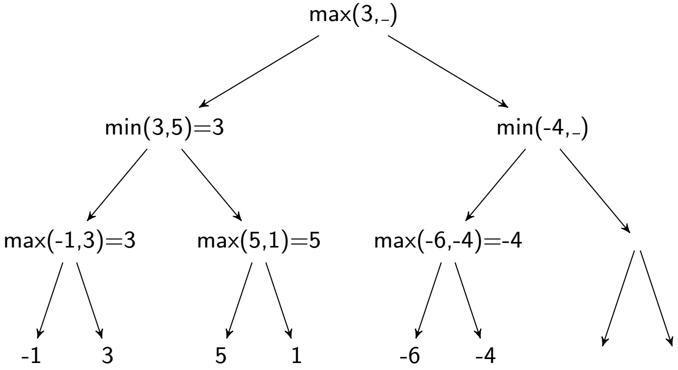
Minimax on an Example



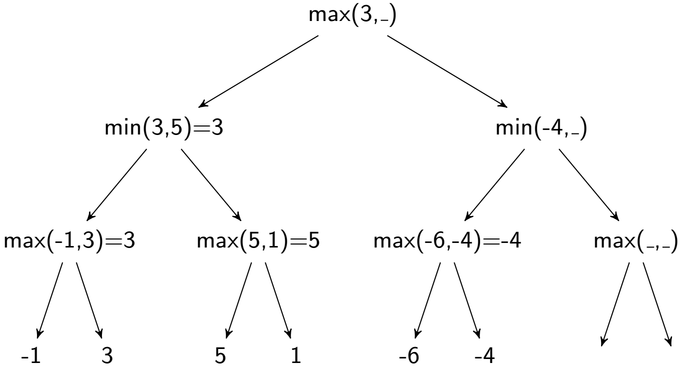
Minimax on an Example



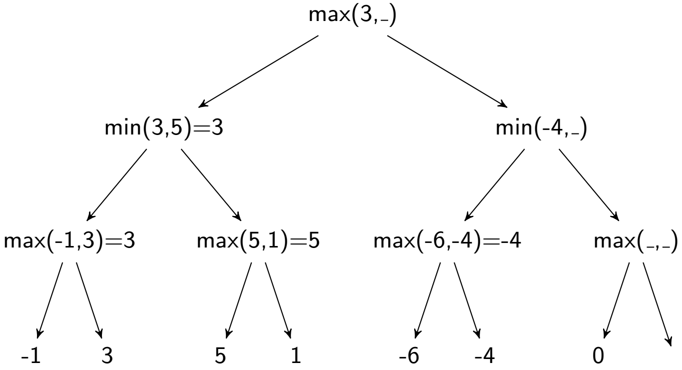
Minimax on an Example



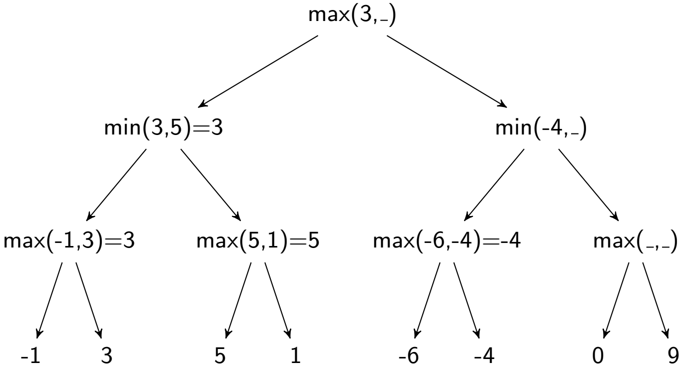
Minimax on an Example



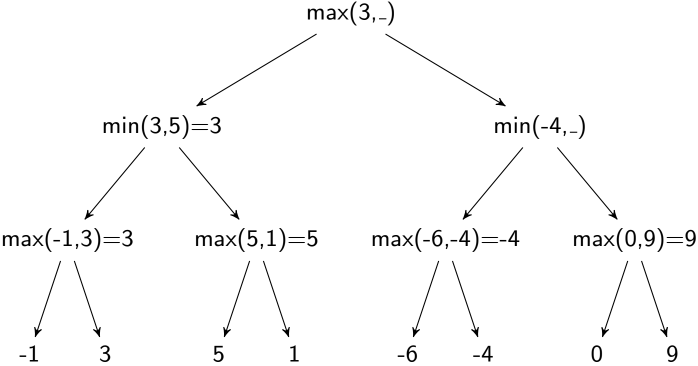
Minimax on an Example



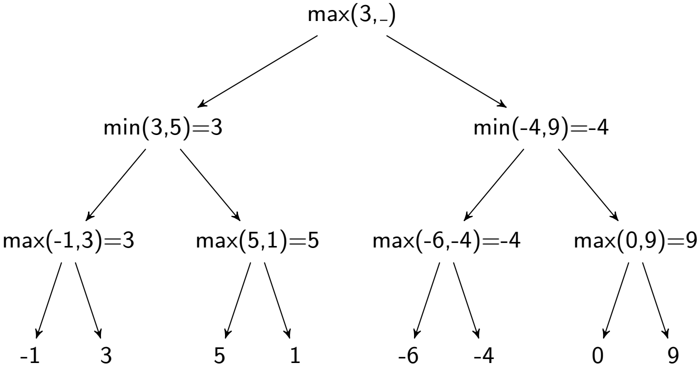
Minimax on an Example



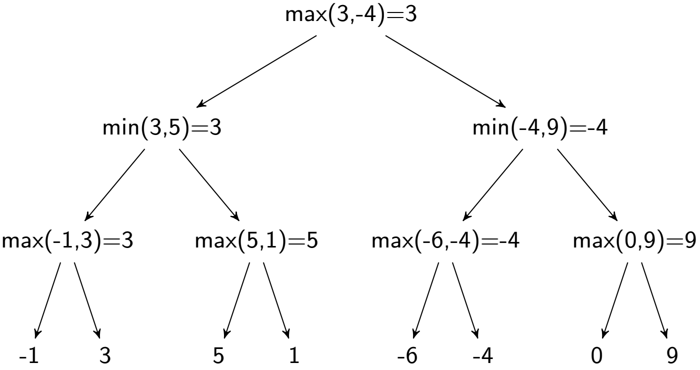
Minimax on an Example



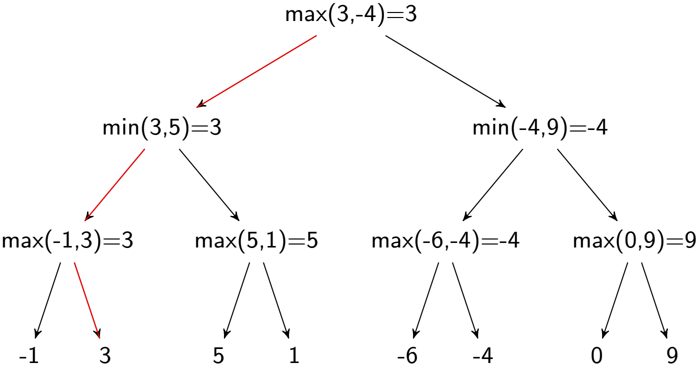
Minimax on an Example



Minimax on an Example



Minimax on an Example



Alpha-beta Pruning

Alpha-beta pruning can make the game tree smaller while still guaranteeing optimal strategies.

Alpha-beta pruning can make the game tree smaller while still guaranteeing optimal strategies.

The recursive Minimax algorithm **passes down** an α and β value to each node:

- α: **lower bound** on what max can achieve when playing through the choice points leading to the current node.
- β: **upper bound** on what min can achieve when playing through the choice points leading to the current node.

The root node has $\alpha = -\infty$ and $\beta = \infty$.

Additionally, the algorithm iteratively updates the value v of each node. Initially $v = -\infty$ for MAX nodes and $v = \infty$ for MIN nodes.

- **Alpha-cut**: If $v \leq \alpha$ in a MIN node, we can prune further search below that node: MAX has a better choice at a previous choice point.
- **Beta-cut**: If $v \geq \beta$ in a MAX node, we can prune further search below that node: MIN has a better choice at a previous choice point.

Alpha-beta Pruning

α-values concern the choice of MAX, β-values concern the choice of MIN.

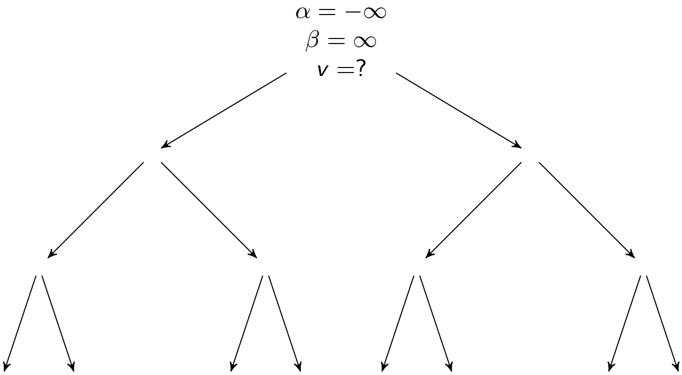
ALPHA-BETA-SEARCH algorithm (see pseudocode in R&N):

- MAX nodes passes α values down:

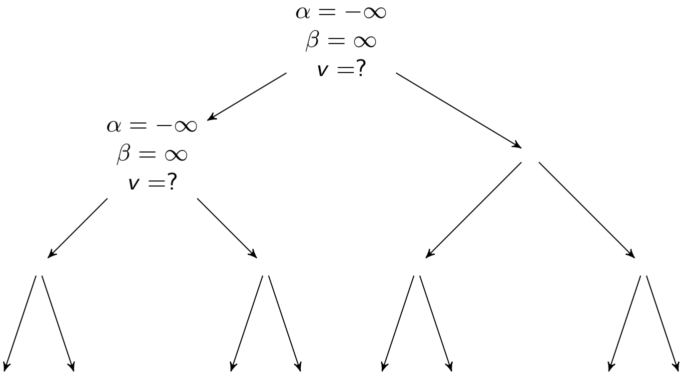
  the max of the α- and v-values of the parent.
- MIN nodes passes β values down:

  the min of the β and v-values of the parent.

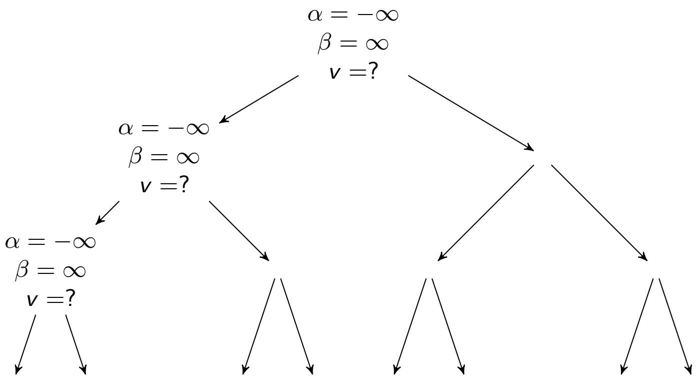
Alpha-beta Pruning: An Example



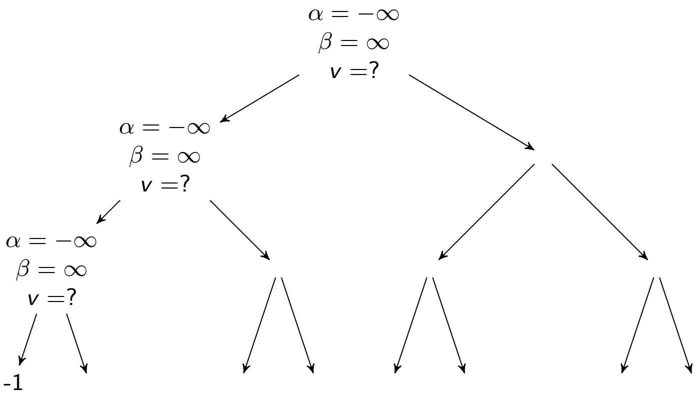
Alpha-beta Pruning: An Example



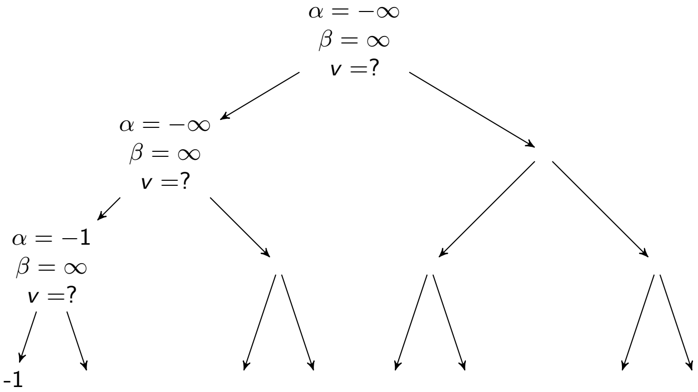
Alpha-beta Pruning: An Example



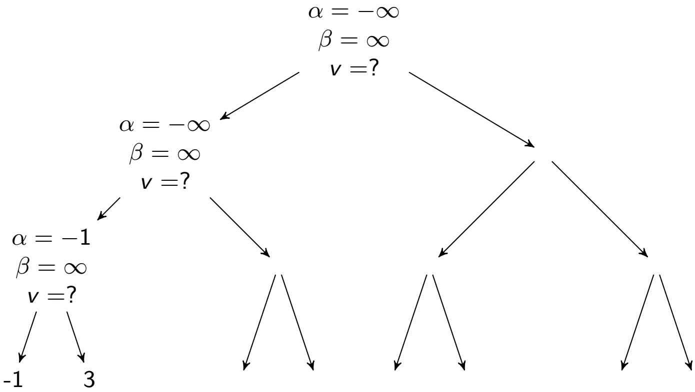
Alpha-beta Pruning: An Example



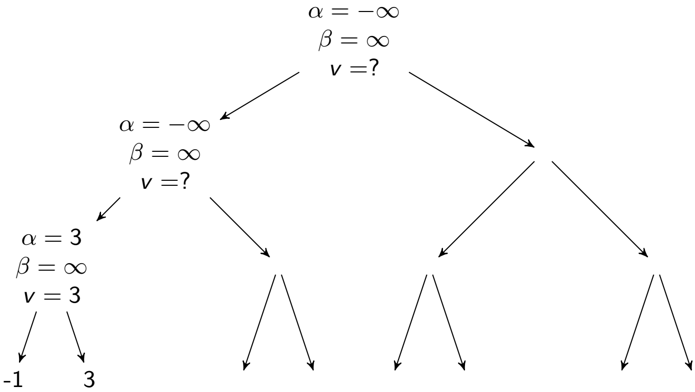
Alpha-beta Pruning: An Example



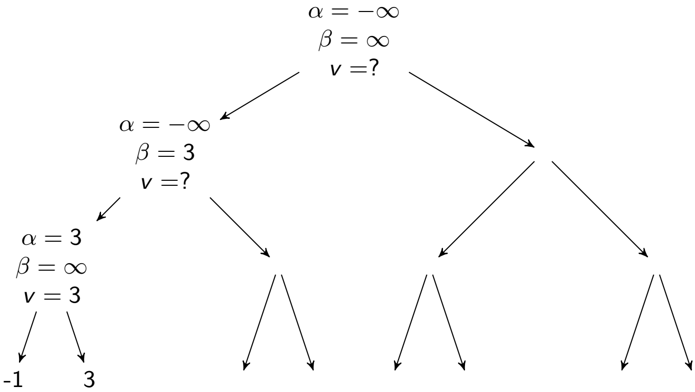
Alpha-beta Pruning: An Example



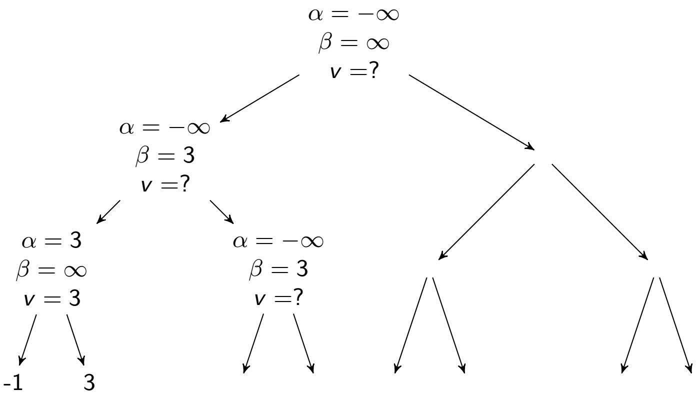
Alpha-beta Pruning: An Example



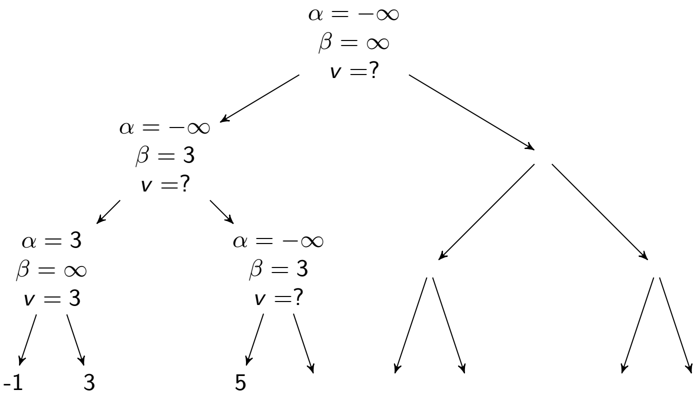
Alpha-beta Pruning: An Example



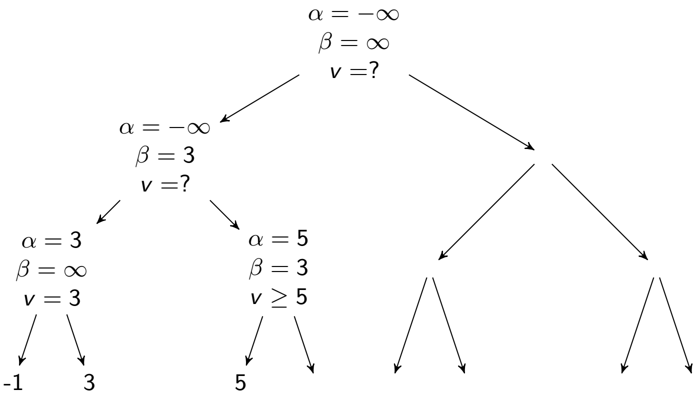
Alpha-beta Pruning: An Example



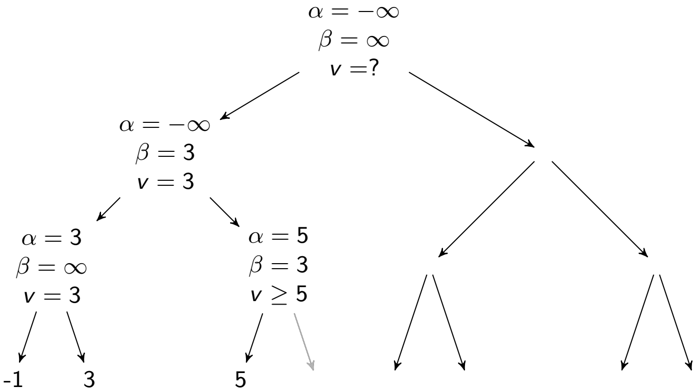
Alpha-beta Pruning: An Example



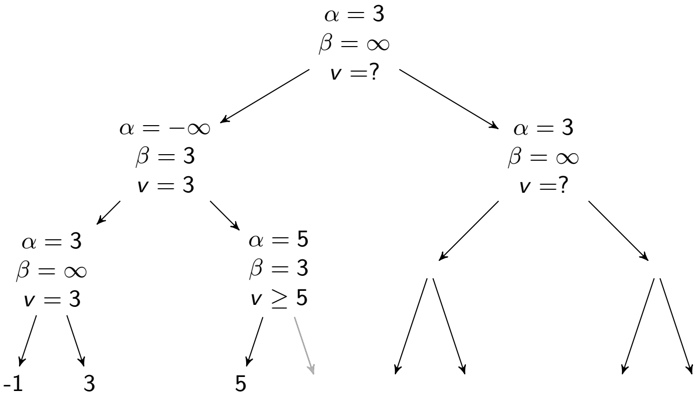
Alpha-beta Pruning: An Example



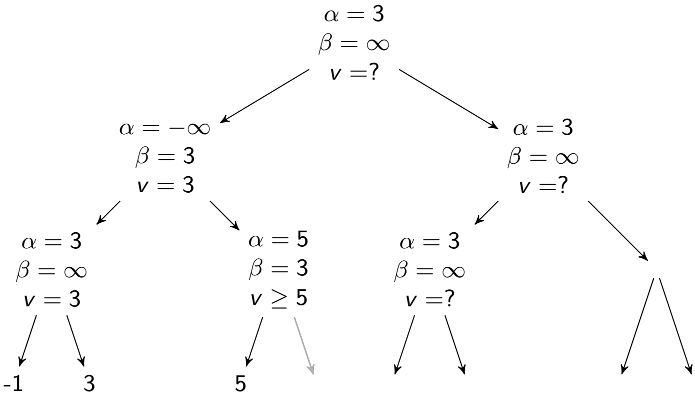
Alpha-beta Pruning: An Example



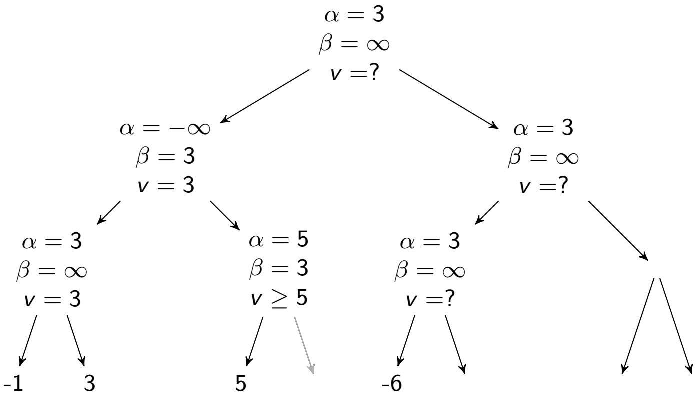
Alpha-beta Pruning: An Example



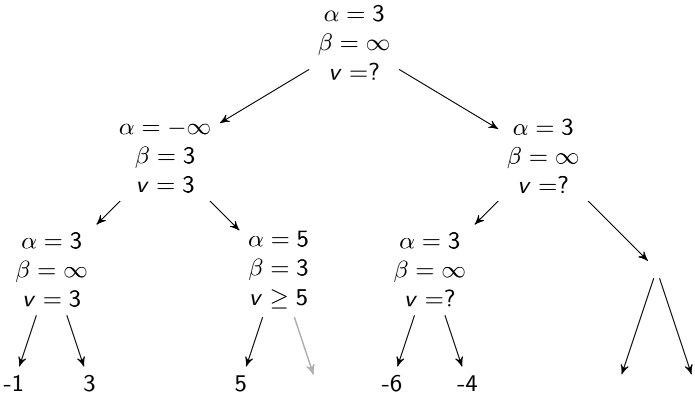
Alpha-beta Pruning: An Example



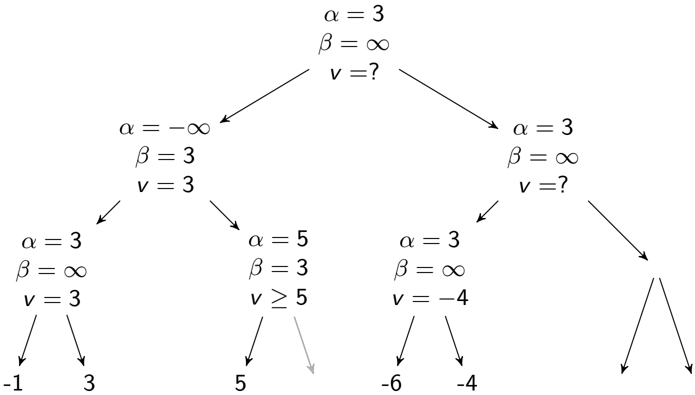
Alpha-beta Pruning: An Example



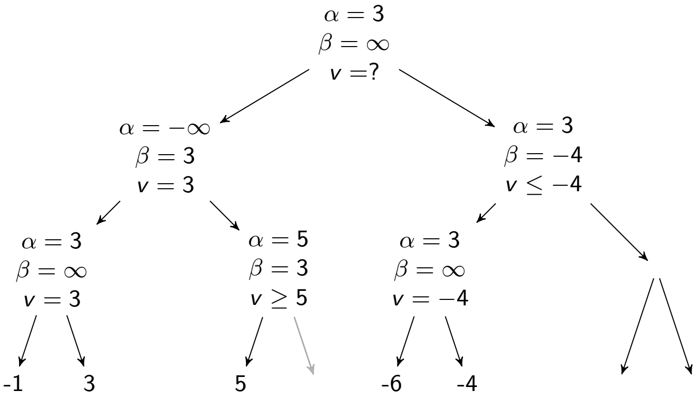
Alpha-beta Pruning: An Example



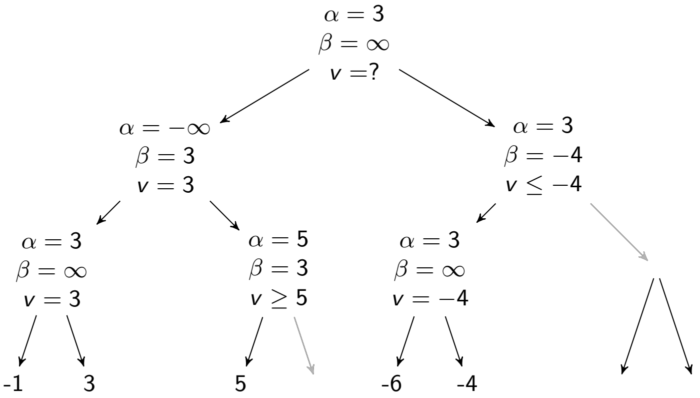
Alpha-beta Pruning: An Example



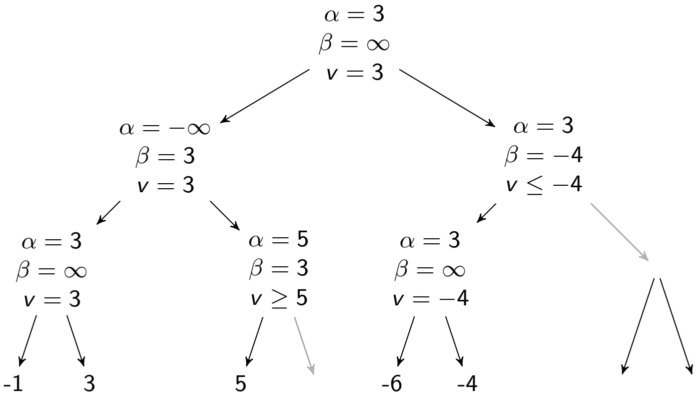
Alpha-beta Pruning: An Example



Alpha-beta Pruning: An Example



Alpha-beta Pruning: An Example



WHEN SEARCHING TO THE END IS NOT POSSIBLE

- Use CUTOFF-TEST(S) instead of TERMINAL-TEST(S). E.g. limit by depth (search only to depth 5).
- Use EVAL(s, p) instead of UTILITY(s, p). This is called an **evaluation function**.

EVAL(s, p): How desirable is state s for player p?
E.g. an estimate of the chance of winning or the expected utility.

Expected utility. If 20% chance of getting utility 5 and 80% chance of getting utility 2, expected utility is $0.2 * 5 + 0.8 * 2 = 2.6$.

When searching to the end is not possible

- Use $\textsc{Cutoff-Test}(\textsc{s})$ instead of $\textsc{Terminal-Test}(\textsc{s})$. E.g. limit by depth (search only to depth 5).
- Use $\textsc{Eval}(s, p)$ instead of $\textsc{Utility}(s, p)$. This is called an **evaluation function**.

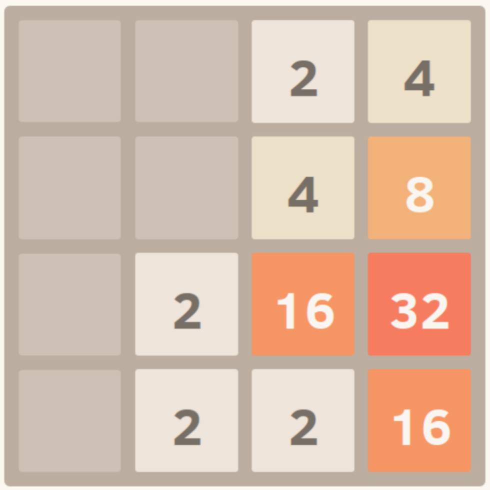
$\textsc{Eval}(s, p)$: How desirable is state s for player p?
E.g. an estimate of the chance of winning or the expected utility.

Expected utility. If 20% chance of getting utility 5 and 80% chance of getting utility 2, expected utility is $0.2 * 5 + 0.8 * 2 = 2.6$.

Evaluation functions in adversarial search play approximately the same role as heuristic functions in classical search: an estimate of "how good" the state is that can be used to guide the search/decisions.

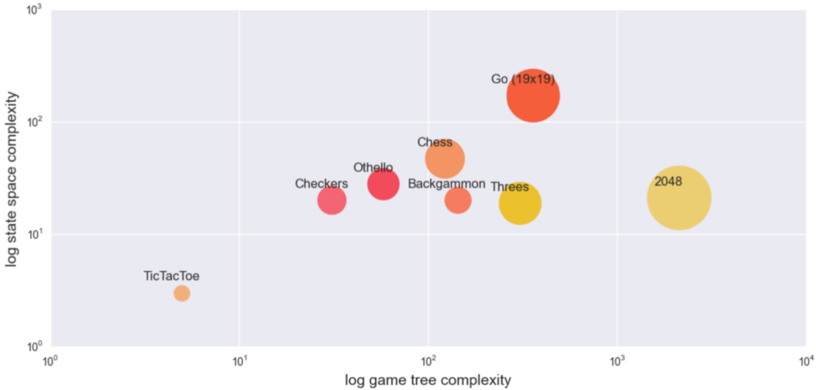
$$\textsc{H-Minimax}(s, d) =$$
$$\begin{cases} \textsc{Eval}(s) & \text{if } \textsc{Cutoff-Test}(s, d) \\ \max_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s, a), d+1) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \min_{a \in \textsc{Actions}(s)} \textsc{Minimax}(\textsc{Result}(s, a), d+1) & \text{if } \textsc{Player}(s) = \textsc{min} \end{cases}$$

Example: AI for 2048



Games and Their Combinatorial Complexity

- State space complexity: number of possible states of the game.
- Game tree complexity: number of possible games.



10^n on these axes mean that the complexity/size is $10^{(10^n)}$.

Stochastic games: Expectimax

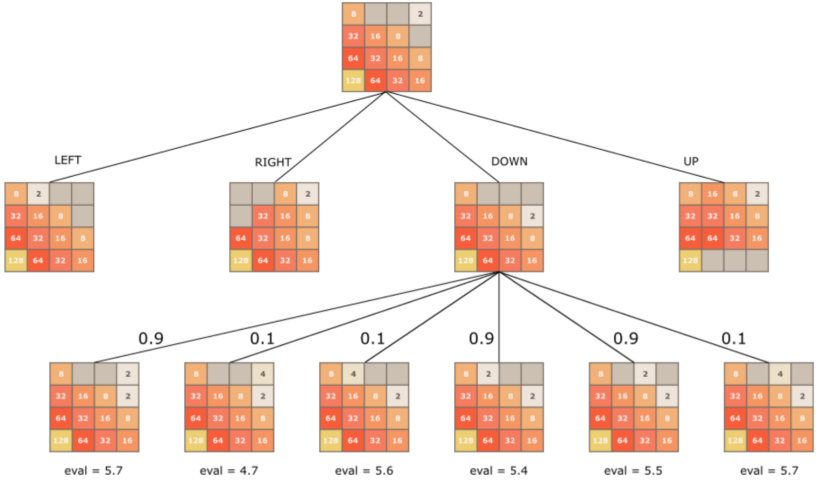
In 2048, the opponent is nature (the game engine), that makes probabilistic decisions, not maximising decisions. We call this player CHANCE. The best suited algorithm is Expectimax rather than Minimax.

Stochastic games: Expectimax

In 2048, the opponent is nature (the game engine), that makes probabilistic decisions, not maximising decisions. We call this player CHANCE. The best suited algorithm is EXPECTIMAX rather than MINIMAX.

$$\textsc{Expectimax}(s) =$$
$$\begin{cases} \textsc{Utility}(s, \textsc{max}) & \text{if } \textsc{Terminal-Test}(s) \\ \max_{a \in \textsc{Actions}(s)} \textsc{Expectimax}(\textsc{Result}(s, a)) & \text{if } \textsc{Player}(s) = \textsc{max} \\ \Sigma_{a \in \textsc{Actions}(s)} P(a, s) \cdot \textsc{Expectimax}(\textsc{Result}(s, a)) & \text{if } \textsc{Player}(s) = \textsc{chance} \end{cases}$$

Here $P(a, s)$ is the probability of CHANCE chosing action a in s.



Expectimax value of going down from root state:

$$\frac{0.9 \cdot (5.7 + 5.4 + 5.5) + 0.1 \cdot (4.7 + 5.6 + 5.7)}{3} = 5.512.$$

Demo of Expectimax on 2048

Expectimax for 2048 (by a previous student Kristine Strandby):

http://kstrandby.github.io/2048-Helper/

THE END OF LECTURE 5